PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-216845
(43)Date of publication of application : 27.08.1993

(51)Int.Cl. G06F 15/16
G06F 13/00

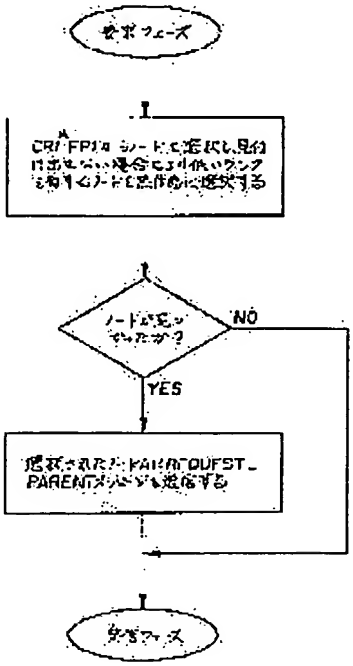
(21)Application number : 04-262099
(22)Date of filing : 30.09.1992

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : ALLON DAVID
BACH MOSHE
MOATTI YOSEF
TEPERMAN ABRAHAM

(30)Priority
Priority number : 91 99923 Priority date : 31.10.1991 Priority country : IL

(54) METHOD FOR OPERATING COMPUTER IN NETWORK

(57)Abstract:
PURPOSE: To balance a load as quickly as possible.
CONSTITUTION: A computer forms a logical link with another computer in a network so that a tree structure can be formed, logically links with one computer in a tree higher order, and logically links with the computer in a tree lower order. Information related with the present load of the computer and the loads of at least several computers among the other computers is held in the computer, this information is periodically distributed by the computer in which the information is stored to the plural computers logically linked with the computer, the information of itself is updated according to the information, and the computer in the network which can receive the excessive load is judged.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平5－216845

(43)公開日　平成５年(1993)８月27日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　15/16	３８０　Ｚ	9190－5L		
13/00	３５５	7368－5B		

審査請求　有　　請求項の数９(全 15 頁)

(21)出願番号　特願平4－262099

(22)出願日　平成４年(1992)９月30日

(31)優先権主張番号　９９９２３

(32)優先日　1991年10月31日

(33)優先権主張国　イスラエル（ＩＬ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ダビッド　アロン
イスラエル　エルサレム　メイヤー　ナカル　ストリート　49／8

(74)復代理人　弁理士　谷　義一　（外３名）

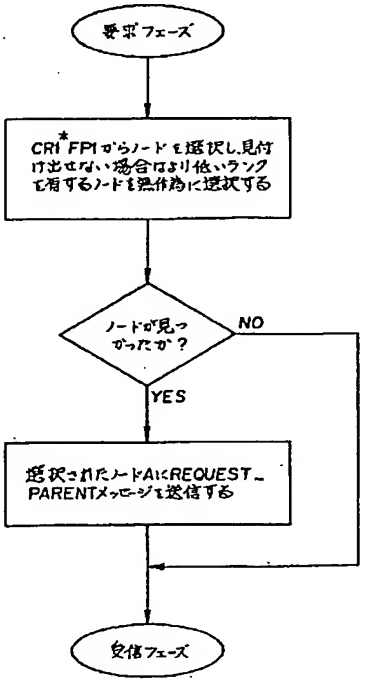
最終頁に続く

(54)【発明の名称】　ネットワーク内のコンピュータを操作する方法

(57)【要約】

【目的】　より速く負荷を平衡させる。

【構成】　ツリー構造が形成されるように、当該コンピュータはネットワーク内の他のコンピュータと論理リンクを形成し、ツリー上位の１つのコンピュータに論理的にリンクし、そのツリー下位のコンピュータに論理的にリンクする。当該コンピュータの現在負荷と、前記他のコンピュータのうちの少なくとも幾つかのコンピュータの負荷とに関する情報をコンピュータ内に保持し、そのコンピュータに論理的にリンクされた複数のコンピュータに、前記情報が記憶されたコンピュータが前記情報を周期的に配送し、かつ、その情報に従ってそれ自体の情報を更新し、その情報を用いて、余分の負荷を受け取ることができるネットワーク内のコンピュータを判定する。



1

【特許請求の範囲】

【請求項１】　ツリー構造が形成されるように、ネットワーク内の当該コンピュータと他のコンピュータの間に論理リンクを形成し、前記当該コンピュータはツリーの上位の１つのコンピュータに論理的にリンクされるとともに、そのツリーの下位のコンピュータに論理的にリンクされるステップと、

前記ネットワーク内の前記当該コンピュータの現在負荷と、前記他のコンピュータのうちの少なくとも幾つかのコンピュータの負荷とに関する情報であって、かつ記憶された情報を、前記コンピュータに論理的にリンクされた複数のコンピュータに周期的に配送し、前記複数のコンピュータから同様のこのような情報を受信し、かつ、その情報に従ってそれ自体の情報を更新することにより、前記情報を前記コンピュータに保持し、その結果、その情報を用いて、余分の負荷を受け取ることができるネットワーク内のコンピュータを判定するステップとを備えたことを特徴とするネットワーク内のコンピュータを操作する方法。

【請求項２】　前記コンピュータが過負荷になると、前記情報を用いて、余分の負荷を受け取ることができるコンピュータを判定し、そのコンピュータに少なくとも１つのタスクを伝送するステップを備えたことを特徴とする請求項１に記載のネットワーク内のコンピュータを操作する方法。

【請求項３】　前記コンピュータに論理的にリンクされているツリー上位のコンピュータに障害が発生するか、あるいは動作不能になった場合、新しい下方リンクを受け取る容量を有するネットワーク内の別のネットワークに新しく論理リンクを生成するステップを備えたことを特徴とする請求項１または２に記載のネットワーク内のコンピュータを操作する方法。

【請求項４】　前記コンピュータにより負荷情報を周期的に配送して、前記コンピュータにリンクされている前記ツリー内のコンピュータが動作可能か否かを判定し、前記負荷情報の周期的な配送が可能であり、かつ、前記コンピュータがリンクされているツリーの下位のコンピュータの１つが動作不可能な場合、そのコンピュータは新しい下方リンクを受け取る容量を有するものとして印を付けるステップを備えたことを特徴とする請求項１ないし３のいずれかに記載のネットワーク内のコンピュータを操作する方法。

【請求項５】　前記コンピュータに記憶された情報は多くの項目を含み、各項目は

（ｉ）前記ネットワーク内の前記コンピュータのうちの１つのコンピュータの負荷、

（ｉｉ）コンピュータとコンピュータを分離するツリー内のリンクの数、

（ｉｉｉ）前記コンピュータにリンクされ、前記項目が最後に受信される前記コンピュータのうちの１つのコンピュータとに関する情報を含み、かつ、

2

前記コンピュータがそのコンピュータにリンクされているコンピュータから、同様の情報を受信すると、

（ａ）受信された情報の各項目内のリンクの数を１だけインクリメントするステップと、

（ｂ）前記コンピュータから発信され、かつ、受信された情報内の項目を削除するステップと、

（ｃ）送信するコンピュータから受信されたコンピュータ内に既に記憶されている情報の項目を削除するステップと、

（ｄ）ローカルベクトルと受信されたベクトルの両方に現れる項目（コンピュータ）の距離を比較するステップと、

１．ローカル項目の距離が受信された項目の距離より大きいか、あるいは等しい場合、ローカル項目を削除する。

２．受信された項目の距離がローカル項目の距離より大きいか、あるいは等しい場合、その受信された項目を削除する。

（ｅ）その受信された項目を既に前記コンピュータに記憶されている情報と併合するステップと、

（ｆ）併合された情報を負荷の昇順に分類し、等しい負荷を有する項目を、前記コンピュータからのリンク分離数の昇順で分類するステップとを実行することを特徴とする請求項１ないし４のいずれかに記載のネットワーク内のコンピュータを操作する方法。

【請求項６】　情報をそれぞれ配送する前に、同一の負荷を有するコンピュータに関する情報の項目と、前記コンピュータからのリンク分離数とを無作為に置換することを特徴とする請求項５に記載のネットワーク内のコンピュータを操作する方法。

【請求項７】　前記コンピュータに対応する項目が最初に前記コンピュータにより分類された情報に現れた場合、カウンタが前記項目に接続され、かつ、前記コンピュータの負荷に初期設定され、その項目が依然最初であり、情報が配送された場合はいつでも、前記カウンタはインクリメントされ、前記カウンタが正になった場合、その項目は前記情報から取り外し、そのことにより、異なるコンピュータに記憶された情報に同一の項目が最初に現れる確率を小さくすることを特徴とする請求項５または６に記載のネットワーク内のコンピュータを操作する方法。

【請求項８】　同様のこのようなコンピュータのネットワークで操作することができるコンピュータであって、コンピュータがツリー構造で論理的にリンクされたネットワーク内のコンピュータを識別する識別手段と、

前記コンピュータ上の負荷と前記ネットワーク内の他のコンピュータの少なくとも幾つかに関する情報を記憶する記憶手段と、

前記コンピュータが論理的にリンクされたコンピュータ

3

に前記情報を送信する送信手段と、
前記コンピュータが論理的にリンクされているコンピュータから同様の情報を受信し、かつ、前記記憶された情報を、受信された情報に従って更新する更新手段と、
余分の負荷容量を有するネットワーク内の他のコンピュータの１つを前記情報から選択し、かつ、選択されたコンピュータにタスクを転送する転送手段とを備えたことを特徴とするコンピュータ。
【請求項９】　前記ネットワーク内の全てのコンピュータのリストを記憶するか、あるいはアクセスする手段と、
前記リストから代表の隣接コンピュータとして１つのコンピュータを選択し、前記ツリー構造にリンクする手段と、
前記選択されたコンピュータにメッセージを送信し、リンク要求を示す手段と、
前記選択されたコンピュータから肯定応答が受信された場合、前記識別手段を更新することにより、前記選択されたコンピュータとの論理的なリンクを確立する確立手段と、
他のコンピュータからのメッセージであって、リンクが要求されていることを示すメッセージを受信する受信手段とこのようなリンク要求メッセージを受信したとき、下方リンクを確立するだけの容量があるか否かを判定し、このような容量が存在する場合、前記リンク要求メッセージの送信元に肯定応答を送信し、これに従って、前記識別手段を更新する手段とを備えたことを特徴とする請求項８に記載のコンピュータ。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、ネットワークのコンピュータを動作させる方法に関し、かつ、ネットワークで動作させることができるコンピュータに関する。
【０００２】
【従来の技術】コンピュータのＬＡＮ（ｌｏｃａｌ　ａｒｅａ　ｎｅｔｗｏｒｋ）は、多くの組織では欠くことのできないものである。コンピュータの大きなネットワークの問題の１つとして、ネットワークのコンピュータを全て効率的に利用するには困難なことがあるということである。負荷分割すなわち平衡方法では、全てのコンピュータをビジー状態にしようとすることにより、システムスループットがますます高められる。この平衡方法は過負荷状態のコンピュータからプロセスをオフロードしてコンピュータをアイドル状態にすることにより実施され、よって、全ての計算機上の負荷が等化され、全体的な応答時間を最小にする。
【０００３】負荷平衡法は平衡を達成するために用いられる方法に従って分類される。それらは、「静的」、「動的」、「集中型」、および「分散型」に分類される。

4

【０００４】静的負荷平衡法では、決定的または確率的にかかわらず、現行のシステムの状態とは独立に固定的な方策に従う。静的負荷平衡法はインプリメントするには簡単であり、かつ、待ち行列モデルを用いて分析するには容易である。しかし、その潜在的な利点は限定されている。というのは、静的負荷平衡法では大域的なシステム状態の変化を考慮に入れていないからである。例えば、コンピュータは既に過負荷状態にあるコンピュータにタスクを移行させることができる。
【０００５】動的負荷平衡法では、そのシステムは大域的なステータスの変化に注意し、コンピュータの現在の状態に基づいてタスクを移行させるか否かを判定する。動的負荷平衡法は、静的負荷平衡法より本来複雑である。というのは、動的負荷平衡法は、システム内の他のコンピュータの状態を知るコンピュータを必要とするからである。他のコンピュータの状態の情報は絶えず更新しなければならない。
【０００６】集中型負荷平衡法では、１つのコンピュータは大域的な状態情報を含み、全ての判定を行う。分散型負荷平衡法では、大域的な情報を有するコンピュータは１台もない。各コンピュータは、当該コンピュータが有する状態情報に基づき、移行するか否かの判定を行う。
【０００７】負荷平衡法はさらに「送信元開始型」および「受信先開始型」に分類される。送信元開始型方策では、過負荷状態のコンピュータは負荷の軽いコンピュータをシークする。受信先開始型方策では、負荷が軽いコンピュータがタスクを受信する能力を広告（ａｄｖｅｒｔｉｓｅ）する。タスク転送コストが送信元開始型および受信先開始型のもとで同等である場合、送信元開始型方策はシステム負荷を加減するため、受信先開始型方策をアウトパフォームする。
【０００８】通常、動的負荷平衡機構は次の３つのフェーズよりなる。
【０００９】１．ローカル計算機の負荷を測定する。
【００１０】２．ローカル負荷情報をネットワークの他の計算機と交換する。
【００１１】３．選択された計算機にプロセスを転送する。
【００１２】ローカル負荷は、実行待ち行列、記憶装置利用率、ページングレート、ファイル使用、および入出力レート、または他の資源利用率のジョブの数の関数として計算することができる。実行待ち行列の長さは通常受信される負荷メトリックである。
【００１３】他のノードから負荷情報を受信することによりシステム負荷のスナップショットが与えられる。この情報を得ると、各コンピュータは転送方策を実行し、タスクを局所的に実行するか、あるいはタスクを別のノードに転送するかを判定する。判定した結果、タスクを転送する場合は、ロケーション方策が実行され、タスク

5

を転送すべきノードを決定する。

【００１４】多くのコンピュータを有するネットワークで、負荷平衡法が効率的である場合、負荷平衡法は、ネットワークトラヒックがネットワークのサイズに対して線形的に増加するという意味で拡張可能でなければならないし、コンピュータをネットワークに付加したりネットワークから除去したりすることが容易にできるように柔軟性がなければならないし、１つ以上のコンピュータに障害が生じた場合頑強でなければならないし、コンピュータのクラスタ化をある程度サポートできなければならない。

【００１５】集中化方法は、簡単であるので魅力的ではあるが、フォールトトレラントな方法ではない。中央コンピュータは死んだノードを検知することができ、かつ、１つまたは２つのメッセージでコネクションを再確立する。しかし、中央ノードに障害が生じた場合は、その方式全体がだめになる。各ノードで優先権を与えられた別の管理者のリストを保持するか、あるいは選挙法をインプリメントして新しく中央ノードになるノードを選挙することにより、負荷平衡を再構成することができる。中央ノードに障害が生じると、他のノードは新しい中央ノードに切り換える。しかし、このようにすると、管理がますます複雑になり、しかも、その方法のコストが高くなる。

【００１６】さらに、中央化方法に対する管理オーバヘッドは、大きすぎて受託することができない。ノードの数が増加するつれて、中央のノードが負荷情報を処理するのに費やす時間が増加し、ついには、中央ノードが過負荷状態になるに違いない。従来の負荷平衡方法には動的および分散型であるものが幾つかある。

【００１７】例えば、Ｂａｒａｋ　Ａ．ａｎｄ　Ｓｈｉｌｏｈ　Ａ．　ＳＯＦＴＷＡＲＥ－ＰＲＡＣＴＩＣＥ　ＡＮＤ　ＥＸＰＥＲＩＥＮＣＥ，１５（９）：９０１－９１３，Ｓｅｐｔｅｍｂｅｒ　１９８５［Ｒ１］に記載された方法では、固定されたサイズの負荷ベクトルであって、かつ、コンピュータ間に周期的に配送される負荷ベクトルを、各コンピュータが保持している。各コンピュータは次の方法を実施する。まず、コンピュータはそのコンピュータ自体の負荷値を更新する。ついで、そのコンピュータは無作為に１つのコンピュータを選択し、選択されたコンピュータに、そのコンピュータの負荷ベクトルの半分の一方を送信する。負荷ベクトルを受信すると、各コンピュータは、予め規定された規則に従って、情報をローカル負荷ベクトルと併合する。

【００１８】

【発明が解決しようとする課題】この方法の問題点は、負荷ベクトルのサイズを注意深く選択しなければならないことである。負荷ベクトルに対する適正な値を見つけ出すことは困難であり、任意のシステムに適合しなければならない。大きい負荷ベクトルは多くのノードに対する負荷情報を有する。従って、ノードは負荷の軽いノードを知って、移行する多くのプロセスを直ちに受信するという機会が増加し、過負荷になるものが多い。一方、負荷ベクトルのサイズは適正な時間でネットワークを伝播できないほど小さくすべきでない。ネットワークのｎ個のノードに対して、負荷情報をネットワークを介して伝播する予期した時間は０（ｌｏｇ　ｎ）である。

【００１９】Ｒ１に記載された方法は、負荷ベクトルの長さを調整した後任意の数のコンピュータを処理することができる。そのため、その方法は拡張が可能である。しかし、柔軟性のみがある程度ある。小数のノードをネットワークに付加するには、全てのコンピュータの負荷ベクトルのサイズを変更する必要がある。従って、管理オーバヘッドを増加する。その方法はフォールトレラントがあり、単一の障害にも動作し続ける能力がある。しかし、デッドノードを検知し、かつ、負荷ベクトル内のデッドノードに関する情報を更新するための組み込み機構がない。従って、障害が生じたノードの情報は伝播されないが、他のノードはプロセスを障害が生じたノードに移行し続けることができる。その結果、応答時間が減少する。

【００２０】単位時間当たりの通信数は０（ｎ）である。しかし、その方法はクラスタ化をサポートしない。１つのノードか他のノードにプロセスをオフロードすると、そのノードはそのノードの負荷ベクトルから対象を選択する。ノードｐの負荷ベクトル内のノードに関する情報が無作為のノードから更新されるので、オフロードする対象の集合は無作為のｎの部分集合であって、制御されたｐの集合ではない。

【００２１】Ｌｉｎ　Ｆ．Ｃ．Ｈ．ａｎｄ　Ｋｅｌｌｅｒ　Ｒ．Ｍ．　ＰＲＯＣＥＥＤＩＮＧＳ　ＯＦ　ＴＨＥ　６ＴＨ　ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＣＯＮＦＥＲＥＮＣＥ　ＯＮ　ＤＩＳＴＲＩＢＵＴＥＤ　ＣＯＭＰＵＴＩＮＧ　ＳＹＳＴＥＭＳ，３２９－３３６，Ｍａｙ　１９８６［Ｒ２］には、傾斜モデルを用いた分散型および同期負荷平衡方法が開示されている。コンピュータがグリッドに論理的に配置され、コンピュータの負荷が表面として表されている。

【００２２】各コンピュータはグリッド上の隣接するコンピュータとのみ対話する。負荷の大きいコンピュータは「丘」であり、アイドル状態のコンピュータは「谷」である。中性コンピュータは過負荷状態でもなくアイドル状態でもないコンピュータである。負荷平衡はその表面の弛張の一形態である。タスクは丘から谷に移行し、表面が平らになる。各コンピュータは最も近いアイドル状態のコンピュータまでの距離を計算し、その距離を用いてタスクを移行させる。

【００２３】過負荷状態のノードはタスクをアイドル状態のノードの方向にある近接するコンピュータに転送する。タスクはアイドル状態のノードに到達するまで移動

し続ける。タスクが到達したとき、最初のアイドル状態のノードが過負荷状態にされた場合、そのタスクは別のアイドル状態のノードに移動する。

【００２４】この方法は拡張可能であるが、部分的にのみ柔軟性がある。グリッドの縁にノードを付加したり、グリッドの縁のノードを取り除いたりすることは容易であるが、グリッドの真中にノードを付加したり、その真中のノードを取り除いたりするのは困難である。というのは、そのグリッドが固定されているからである。ノードを付加したり取り除いたりすると、再構成が必要である。その再構成によりその方法が備えているオーバヘッドが増加する。死んだノードを検知することは容易ではない。ノードの状態の変化のみが近接するコンピュータに送信されるので、ノードの障害は、ジョブが近接するコンピュータに転送されるまで、検知されないままである。検知が遅れるとプロセスの移行が遅らされ、そのため、全体的な応答時間が増加する。この方法では、クラスタ化はサポートされていない。異なる方向に転送することができる近接するコンピュータのうちの１つに、各ノードはプロセスを転送する。過負荷状態のノードは、オフロードされたプロセスを結果的にどこで実行するかは制御しない。そのグリッドを開始するための管理オーバヘッドは小さく、メッセージの数は０（ｎ）である。再構成に対する管理のオーバヘッドは大きい。というのは、障害の生じたノードに対して近接するノードが変化する必要があるからである。この方法では、ノードオーバヘッドは大きく、ネットワークのトラヒックが増加する。というのは、タスクはアイドル状態のノードに直接移動するというよりは、ホップして移動し、しかも、プロセスの移行に時間がかかるからである。

【００２５】Ｓｈｉｎ Ｋ．Ｇ．ａｎｄ Ｃｈａｎｇ Ｙ．Ｃ． ＩＥＥＥ ＴＲＡＮＳＡＣＴＩＯＮ ＯＮ ＣＯＭＰＵＴＥＲＳ，３８（８），Ａｕｇｕｓｔ １９８９［Ｒ３］にて提案されている仲間集合（ｂｕｄｄｙ ｓｅｔ）アルゴリズムは、負荷分割を非常に速く行うことを目的としている。

【００２６】各ノードに対して２つのリストが規定されている。仲間集合、すなわち、その近接するコンピュータの集合と、その好ましいリスト、すなわち、タスクが転送される仲間集合内の順序付けされたノードのリストである。各ノードは次の状態のうちの１つの状態になることができる。すなわち、標準より小さい負荷がかかった状態（ｕｎｄｅｒ－ｌｏａｄｅｄ）、中ぐらいの負荷がかかった状態（ｍｅｄｉｕｍ－ｌｏａｄｅｄ）、過負荷状態である。各ノードは仲間集合内の全てのノードのステータスを含む。ノードのステータスが変化した場合はいつでも、ノードは新しい状態をその仲間集合の全てのノードにブロードキャストする。好ましいリストの内部順序は、仲間集合内の２つのノードが同一のノードにタスクを転送する確率を最小にするためノードごとに変

更する。１つのノードが過負荷状態になると、そのノードはその好ましいリストを走査して標準より小さい負荷がかかったノードを探し、そのノードにタスクを転送する。過負荷状態のノードは好ましいリストから外される。仲間集合およびそれらの好ましいリストはオーバラップしており、そして、プロセスはそのネットワークを巡回する。

【００２７】再度述べるが、この方法は、メッセージの数がネットワークｎのコンピュータの数と線形的に増加する点では拡張可能であるが、柔軟性がない。ネットワークにノードを付加したり、ネットワークからノードを取り除くには、関係する仲間集合の全てのノード内の好ましいリストを再計算する必要がある。１つのノードか２つ以上の仲間集合に付加された場合は、各仲間集合の各ノードに対して再計算しなければならない。Ｒ２に記載された傾斜モデルと同様の理由で死んだノードを検知することは困難である。クラスタ化はサポートされているが、再構成はコストがかかる。というのは、ノードを付加したり取り除いたりするため、ノードは新しい仲間集合と好ましいリストを再計算する必要があるからである。そのため、その方法が備えた管理オーバヘッドは大きい。

【００２８】

【課題を解決するための手段】本発明に係る第１の実施態様では、コンピュータネットワークのコンピュータを走査する方法を提供する。その方法は、そのコンピュータとネットワーク上の他のコンピュータとがツリー構造を形成するように論理リンクを生成し、そのコンピュータはそのツリーの上位の１つのコンピュータと下位の多くのコンピュータとに論理的にリンクされ、前記ネットワーク内の前記当該コンピュータの現在負荷と、前記他のコンピュータのうちの少なくとも幾つかのコンピュータの負荷とに関する情報であって、かつ記憶された情報を、前記コンピュータに論理的にリンクされた複数のコンピュータに周期的に配送し、前記複数のコンピュータから同様のこのような情報を受信し、かつ、その情報に従ってそれ自体の情報を更新することにより、前記情報を前記コンピュータに保持し、その結果、その情報を用いて、余分の負荷を受け取ることができるネットワーク内のコンピュータを判定することができる。

【００２９】本発明は送信元開始型および受信先開始型負荷平衡法に適用可能である。送信先開始型負荷平衡方式を用いた本発明の実施例は、コンピュータが過負荷状態になると、その情報を用いて、余分の負荷を受け取ることができるコンピュータを判定し、そのコンピュータに少なくとも１つのタスクを転送するステップをさらに含む。

【００３０】上述する方法を用いて、各コンピュータを操作することにより、コンピュータネットワークを操作する方法をさらに提供する。

9

【００３１】ランクを各コンピュータに割り当てることにより、論理リンクを生成することができる。どの２つのコンピュータにも同一のランクが割り当てられない、各コンピュータはそれより下位のランクの１つのコンピュータと、それより上位のランクの多くのコンピュータに論理的にリンクし、ツリーを形成する。

【００３２】コンピュータに障害が生じるか、あるいは操作不能になった場合、障害が発生したコンピュータがリンクされているツリーの下位の各コンピュータと、新しい下位のリンクを受け取る容量を有する他のコンピュータとの間に新たに論理リンクを生成することにより、ツリー構造を保守することができる。

【００３３】改善された負荷平衡方式が採用され、この方式の特徴は、各コンピュータの通信負荷と、各コンピュータに記憶するのに必要なネットワークトポロジーの知識を制限することにより拡張するという点にある。従って、各コンピュータのアクションはネットワーク上のコンピュータの総数とは無関係である。

【００３４】ツリー構造は動的に構築されかつ保守されるので、その方法はまた柔軟性がある。ツリー構造を変化させるには、再構成されたコンピュータのランク付けのみを変化させる必要がある。新しいランク付けファイルは、循環することを避けるため、現在のランク付けファイルと一致していなければならない。そのコンピュータが隣接するコンピュータから見て操作不能に見えるように、そのコンピュータを論理的に切断し、その構成ファイルを交換し、かつ、コンピュータをツリーと再コネクションさせることにより、コンピュータが再構成される。ツリーの他のノードを乱したり、ローカルサービスを遮断せずに新たに構成される。

【００３５】また、障害が生じたノードを検知し、かつ、そのノードを再コネクションすることが提供されている。ツリーの上位の１つのコンピュータがある時間内で応答しない場合は、その１つのコンピュータと前もってコネクションされている上位のランクのコンピュータは、そのツリーのどこかと再リンクを試みる。各上位のランクのコンピュータが再リンクに一時的に失敗した場合、各上位のランクのコンピュータは切断されたサブツリーのルートとして動作し、そのサブツリー内で負荷が平衡する。結果的に、既に試みたノードかあるいは死んだノードをマークするフラグはリセットされることになり、ツリーに周期的に接続を試みるコンピュータはそのツリーと再コネクションすることになる。

【００３６】都合のよいことに、負荷情報が各コンピュータによりツリーのリンクを介して配送され、各コンピュータにリンクされたコンピュータが操作可能か否かを判定する。各コンピュータとツリーの上位でリンクされているコンピュータが　操作不可能である場合、各コンピュータはシークし、下位のランクのコンピュータと新たにリンクを生成するとともに、新しい下位のリンクと

10

リンクする能力を有するコンピュータの１つと新たにリンクを生成する。各コンピュータがリンクされたツリーの下位のコンピュータの１つが、新しい下位のリンクを受け取る容量を有するものの、操作不可能である場合、各コンピュータはマークされる。

【００３７】本発明の好ましい形態では、各コンピュータに記憶された情報は多くの項目を含む。各項目はネットワーク内のコンピュータの特定の１つの負荷に関する情報と、その情報が記憶されたコンピュータからｉ個に分離されたツリーのリンクの数ｉと、前記負荷情報が記憶されたコンピュータに論理的にリンクされたコンピュータの名前（すなわち、ランク）を含む。負荷情報が記憶されたコンピュータから前記各項目が最後に受信される。各コンピュータに論理的にリンクされたコンピュータから、前記各コンピュータが同様の情報を受信すると、次のステップが実施される。

【００３８】ａ）受信された情報の各項目のリンク分離値の数は１だけインクリメントされる。

【００３９】ｂ）受信された情報であって、かつ、情報を受信するコンピュータから発信された情報の項目は削除される。

【００４０】ｃ）情報を受信するコンピュータに既に記憶された情報であって、かつ、情報を送信するコンピュータから受信された情報の項目は削除される。

【００４１】ｄ）ローカルベクトルおよび受信されたベクトルとして現れる項目（コンピュータ）のリンク分離数は比較される。

【００４２】１．ローカル項目の数が大きいかあるいは等しい場合、そのローカル項目は削除される。

【００４３】２．受信された項目の値が大きい場合、その受信された項目は削除される。

【００４４】ｅ）受信された項目はその項目を受信するコンピュータに既に記憶された情報と併合される。

【００４５】ｆ）その併合された項目は負荷の昇順で記憶され、同等の負荷を有する項目は、項目を受信するコンピュータからのリンク分離数の昇順で分類される。

【００４６】情報をコンピュータにそれぞれ配送する前に、同一の負荷と、コンピュータからのリンク分離の数とを有するコンピュータに関する情報の項目を無作為に置換することができ、その結果、異なるコンピュータに記憶された情報に同一の項目が最初に現れる確率が減少する。また、そのコンピュータに対応する項目がそのコンピュータの分類された情報に最初に現れる場合、その項目にカウンタを接続し、かつ、初期設定してそのコンピュータの余分な負荷容量を負の値にすることができる。その項目が依然として最初である場合、その情報が配送されるときはいつでも、そのカウンタがインクリメントされる。そのカウンタが正になると、その情報からその項目が除去される。よって、異なるコンピュータに記憶された情報に同一の項目が最初に現れる確率が減少

11

する。

【0047】本発明の好都合の形態では、その情報をツリーの上位にあるコンピュータに周期的に配送する期間は、そのコンピュータで作成される新しいタスクのレートおよび／またはツリー構造に論理的にリンクされたコンピュータで作成される新しいタスクのレートに関係する。1つ以上のノードの負荷が突然大きくなった場合、この情報がネットワークにより速く広がることになるという利点を有する。

【0048】さらに、本発明で提供される方法の利点は、ネットワークの全体的な負荷が絶えず増加している場合は礼儀正しく処理されるという点にある。このような場合、目標コンピュータに対して、ローカル負荷ベクトルが探索されると、候補がより少なくなり、移行数が減少することになる。負荷が全体的に減少すると、ローカル負荷ベクトルの候補がより多くなり、移行が再開される。言い替えると、ネットワーク性能はネットワーク負荷が飽和している間礼儀正しく低下し、クラッシュすることはない。むしろ、全体的な負荷が正常に戻ると、正常な動作が再開される。

【0049】別の態様から見ると、本発明は、同様のこのようなコンピュータのネットワークで動作することができるコンピュータを提供することができる。このコンピュータは、ツリー構造の中でコンピュータが論理的にリンクされるネットワークのコンピュータを識別する手段と、そのコンピュータおよびネットワーク上の少なくとも幾つかの他のコンピュータの負荷に関する情報を記憶する手段と、ツリー構造でそのコンピュータが論理的にリンクされているコンピュータに前記情報を送信する手段と、そのコンピュータが論理的にリンクされている前記コンピュータからの同様の情報を受信し、かつ、記憶された情報をその情報に従って更新する手段と、余分の負荷容量を有するネットワークの他のコンピュータの1つを前記情報を用いて選択し、かつ、選択されたコンピュータにタスクを転送する手段とを備えていることを特徴とする。

【0050】また、そのコンピュータは、ツリーにリンクされる隣接する候補のコンピュータとして1つのコンピュータをこのリストから選択する手段と、選択されたコンピュータにリンク要求を示すメッセージを送信する手段と、選択されたコンピュータから肯定応答が受信された場合、前記識別する手段を更新することにより、選択されたコンピュータとの論理的なリンクを確立する手段と、リンクが要求されていることを示す他のコンピュータからメッセージを受信する手段と、このようなリンク要求メッセージを受信したとき、ツリーの下位でリンクを確立する容量があるか否かを判定し、かつ、このような容量がある場合は肯定応答をリンク要求メッセージの送信元に送信し、したがって、前記識別する手段を更新する手段とを含むことが好ましい。

12

【0051】ツリーの下方のコンピュータからの同様の情報を当該コンピュータが受信すると、この受信に応答して、前記情報がツリーの下方のコンピュータにより配送される。

【0052】ランクを各コンピュータに割り当てることにより、論理リンクが生成され、かつ、任意の2つのコンピュータも同一ランクが割り当てられず、各コンピュータはより下位の1つのコンピュータにリンクされるとともに、より上位の多くのコンピュータにリンクされ、ツリー構造を形成する。

【0053】さらに、このようなコンピュータのネットワークを提供する。

【0054】

【実施例】本発明の実施例を図面を参照して説明する。

【0055】本発明に係る実施例は2つの主要部よりなる。第1の部分はツリー構築と保守である。第2の部分は負荷平衡情報の交換と保守である。

【0056】ツリー構築と保守

ネットワーク上の各コンピュータにはツリー構築に用いる一意のランクが割り当てられる。以下、「親」という語は下位ランクのコンピュータを参照するために用いられ、この下位のコンピュータに対して1つのコンピュータは上位のリンクを形成することができる。「子」という語は上位ランクのコンピュータをいい、この上位ランクのコンピュータに対して1つのコンピュータは下位のリンクを形成することができる。

【0057】各コンピュータは次の（i）～（v）を含む構成ファイルを記憶し、その構成ファイルにアクセスする。

【0058】（i）そのコンピュータが受け取ることができる直系の子孫の最大数。これらの数は子スロットという。

【0059】（ii）コンピュータのリストおよびコンピュータのそれぞれのランク。

【0060】（iii）「好意的な親」ノードの順序付けリストFP。好意的な親のランクは現在コンピュータより下位である。FPリストはコンピュータにより用いられ親ノードを選択する。FPリストは任意に選択される。親が無作為に選択される場合にはFPリストを空にすることができる。FPリストは下位の全てのコンピュータの部分集合を含む。

【0061】（iv）親および子の応答を待つ時間。

【0062】（v）ノードが死んでいるかあるいは生きているかをプローブするために待機する時間。

【0063】コンピュータCRに対する候補の親のリストは、Pの候補範囲というが、可能な親の範囲を限定する。はじめ、各コンピュータのCRはそのコンピュータより低いランクのコンピュータを全て含む。ノードがCRおよびFPにある場合にのみ、そのノードが親として選択される。FPが空であるか、あるいは使い果たした

場合は、ＣＲのメンバーのみが検査される。

【００６４】各コンピュータはツリーを構築するため、次のステップを実施する。要求フェーズがあり、このフェーズでは、１つのコンピュータは別のコンピュータを選択し、親になることを要求する。また、受信フェーズがあり、このフェーズでは、コンピュータは応答を待ち、他のコンピュータからの要求に応答する。

【００６５】要求フェーズ

１．各コンピュータｉは好意的な親のリストＦＰｉ から別のコンピュータを取り出す。ＦＰｉ が空か、あるいは、ＦＰｉ の全ての候補が失敗した場合、それより下位のコンピュータが無作為に選択される。選択されたコンピュータｊがＣＲｉ にない場合は、ＦＰｉ の次のコンピュータが選択されるか、あるいは、別のコンピュータが無作為に取り出される。全てのノードの試みが失敗した場合は、そのコンピュータは受信フェーズに入る。

【００６６】２．ｉは選択されたコンピュータｊにｒｅｑｕｅｓｔ＿ｐａｒｅｎｔメッセージを送信し、かつ、他のコンピュータからのメッセージを待つ。

【００６７】３．代表の親が、ある期間内に応答しない場合、この代表の親は死んだものと印を付けられ、ｉは要求フェーズに再び入る。

【００６８】要求フェーズ

１．ｒｅｑｕｅｓｔ＿ｐａｒｅｎｔメッセージがｍから到達すると、（ａ）ｉが自由子スロットを有する場合、ｉはｍの要求を受信し、ａｃｋメッセージを送信する。

【００６９】（ｂ）ｉが割り当てられた数の子スロットを使い果たした場合、

１）ｉは子を走査して次に規定する前の子を探す。前の子を見つけた場合、

（ａ）その子は死んだものと印を付けられ削除される。

【００７０】（ｂ）子スロットを利用可能にする。

【００７１】（ｃ）ｉはｍにａｃｋメッセージを送信する。

【００７２】前の子が見つからない場合は、

２）ｉは最も高いランクを有する子ｋを見つける。

【００７３】３）ｒａｎｋ（ｋ）＞ｒａｎｋ（ｍ）である場合、

ａ）ｉはｋのかわりにｍを子として受け入れ、ｍにａｃｋメッセージを送信する。

【００７４】ｂ）ｉはｋにｒａｎｋ（ｍ）をパラメタとして含むｄｉｓｅｎｇａｇｅメッセージを送信する。

【００７５】４）ｒａｎｋ（ｋ）＜ｒａｎｋ（ｍ）である場合、

ａ）ｉはコンピュータの中のリストにｒａｎｋ（ｋ）＞ｒａｎｋ（ｍ）を満足する最も低いランクを有するコンピュータｎを置く。

【００７６】ｂ）ｉはｍにｒａｎｋ（ｎ）をパラメタとして含むｎｏａｃｋメッセージを送信する。

【００７７】２．パラメタｒを有するｄｉｓｅｎｇａｇ

ｅメッセージが到達した場合、

ａ）ｉはその親フィールドをクリアする。

【００７８】ｂ）ｉは範囲［ｒａｎｋ（ｉ）－ｌ，ｒ］外のランクを有するコンピュータをｃｒｉ から削除し、新しい親を探し始め、すなわち、要求フェーズにはいる。

【００７９】３．ａｃｋメッセージが代表の親から到達した場合、そのコンピュータは代表の親を真の親として記録する。

【００８０】４．ａ）ｉはその代表の親のフィールドをクリアする。

【００８１】ｂ）ステップ２ｂを実行する。

【００８２】図１はツリー世代の要求フェーズを示す流れ図であり、図２、図３、図４はツリー世代の受信フェーズを示す流れ図である。

【００８３】ツリー世代の例として、ランク０からランク４を有し、かつ、２人の子を受け取ることができる５つのコンピュータを考察する。親は無作為に選択されるため、その方法により図５に示す段階に到達することができる。

【００８４】コンピュータ１はコンピュータ０のみを親として選択することができるが、コンピュータ０は既に２人の子を有する。そのため、その方法は、コンピュータ０の子３とコンピュータ１を置換し、コンピュータ３にパラメタ１を有するｄｉｓｅｎｇａｇｅメッセージを送信する。よって、図６に示す状態になる。

【００８５】ノード３は範囲２から１までの親を探索し、例えば、ノード１を親として取り上げる。その結果得られるツリーは図７に示すようになる。

【００８６】ノードが全て生きており、かつ、前の子がない場合、ツリー世代は１つのツリーで終了する。

【００８７】上述したツリー世代は不平衡ツリーを構築するか、あるいは、１人の子のみを有する多くのノードを有するツリーを構築することができる。しかし、負荷平衡方法はツリーのトポロジーにより影響されるが、その影響は無視できる程度のものであることが分かっている。

【００８８】ツリーの保守は、死んだノードを検知するか、新しいノードを負荷するか、あるいは、リブートされたノードに再コネクションするかするために必要である。ノードに障害が発生し、かつ、ノードの中には保守のために遮断されるものがあるため、ツリーの保守が必要である。本発明に係る実施例では、このような場合、次のようにして処理する。すなわち、

１．各コンピュータは周期的にｕｐｄａｔｅ＿ｕｐメッセージをその親に送信し、ｕｐｄａｔｅ＿ｄｏｗｎメッセージを待つ。

【００８９】２．指定した期間内に応答がない場合、その親は死んだものと印が付けられ、そのコンピュータは新しい親を探す。

15

【００９０】３．ｕｐｄａｔｅ＿ｕｐメッセージが子から到達した場合、そのノードはｕｐｄａｔｅ＿ｄｏｗｎメッセージに応答する。

【００９１】４．指定した期間内にｕｐｄａｔｅ＿ｕｐメッセージが子から到達した場合、その子は「前の」と印が付けられる。

【００９２】５．ｕｐｄａｔｅ＿ｕｐメッセージが前の子から到達した場合、その子に対する前のフラグが送信され、そのノードはｕｐｄａｔｅ＿ｄｏｗｎメッセージに応答する。

【００９３】６．コンピュータがリブートされるか、あるいはそのコンピュータに付加されると、そのコンピュータは親を探し、すなわち、要求フェーズに入る。

【００９４】７．各コンピュータは既に試みたコンピュータと死んだものと印が付けられたコンピュータとにフラグを立てる。これらのフラグは周期的にクリアされ、ＣＲがリセットされ、より下位のランクを有する全てのコンピュータを含む。従って、リブートされたノードを再びプローブすることができる。

【００９５】例として、図８に示すようにａからｒのラベルを付けた１８個のノードのツリーを考察する。

【００９６】ノードｈに障害が発生した場合、予め規定した期間が経過した後、その親ｃはそのノードｈを「前の」と印をつけ、そのノードｈがｕｐｄａｔｅ＿ｕｐメッセージを受信していないので、そのノードｈを無視することになる。ｕｐｄａｔｅ＿ｄｏｗｎメッセージが到達していないので、ノードｋおよびｒは新しい親を探し始める。今、そのネットワークは図９に示すように多くのばらばらのツリーを備える。

【００９７】ノードｋおよびｒはそれらのＦＰの中に新しい親を探す。それらのＦＰが空である場合は、ノードｋおよびｒは無作為に親を探す。その新しいツリーは図１０に示すツリーのようにすることができ、ノードｋはノードｇをその新しい親として持ち、ノードｒはノードｃをその新しい親として持つ。

【００９８】ツリー保守機構と一緒になったツリー世代により、ネットワークのコンピュータが一般的に単一のツリー構造に配置されることになることが確認される。このことがノード障害の場合に真であることを知るため、ノードｋに障害が発生するか、あるいはノードｋが動作不能になっている任意のツリーについて考察してみよう。ノードｋの親はノードｋを「前の」と印を付け、ノードｋの子はノードｋを死んだものと印を付けることになる。ノードｍがノードｋの子だったと仮定しよう。そのため、ノードｍは新しい親を探す。次の３つの可能性がある。

【００９９】１．ノードｍは新しい親を子として受け取るノードを見つけ出す。よって、単一のツリーが形成される。

【０１００】２．ノードｍはｋの親を代表として取り出すが、自由スロットはない。

16

【０１０１】この場合、「前の」と印が付けられたノードｋのスロットが用いられ、ノードｋは死んだものと印が付けられる。再び、単一のツリーが形成される。

【０１０２】３．ノードｍは新しい親を子として受け取るノードを見つけ出さない。

【０１０３】このことは、新しい親を受け取るであろうノード（少なくとも１つ、すなわち、ｍ－１）が既に試みられていると印を付けられるか、あるいは、死んだものと印が付けられるかした場合にのみ起こる。この場合、そのネットワークは単一のツリーを備えるというよりは、一時的に数多くのサブツリーを備える。この状態は、全てのフラグがリセットされた（保守機構のステップ７）ときに終了し、ついで、ノードｍは再び試み、かつ、親を見つけ出すことになる。

【０１０４】上述した理由付けはノードｋの全ての子に対して適用可能である。

【０１０５】情報の交換と保守

実行待ち行列上のジョブの数は適正な負荷メトリックとして受け取られるのが一般的である。実行待ち行列が予め規定したしきい値、すなわち、ｏｖｅｒｌｏａｄ＿ｍａｒｋを超えた場合、ノードは過負荷と考えられる。実際に興味ある量はそのノードがｏｖｅｒｌｏａｄ＿ｍａｒｋをどの程度超えるかということであるので、本発明に係る実施例では、僅かに異なるメトリックが用いられている。本発明に係る実施例では、負荷はｌｅｎｇｔｈ＿ｏｆ＿ｒｕｎ＿ｑｕｅｕｅ－ｏｖｅｒｌｏａｄ＿ｍａｒｋとして規定されている。例えば、ｏｖｅｒｌｏａｄ＿ｍａｒｋA ＝１０であり、かつ、ｏｖｅｒｌｏａｄ＿ｍａｒｋB ＝８であり、かつ、ノードＡおよびＢの実行待ち行列の長さがそれぞれ５および４である場合は、ｌｏａｄA ＝－５であり、かつ、ｌｏａｄB ＝－４である。したがって、ノードｂの負荷はノードＡの負荷より大きい。

【０１０６】このメトリックが負荷に対して用いられる次の場合や、負荷という語が用いられる場合にはいつでも、ｌｅｎｇｔｈ＿ｏｆ＿ｒｕｎ＿ｑｕｅｕｅ－ｏｖｅｒｌｏａｄ＿ｍａｒｋの意味である。

【０１０７】ネットワークの各コンピュータは、そのネットワークの他のコンピュータ上の情報を保持する、分類されたベクトルを記憶する。そのベクトルの各項目は次のものを含む。

【０１０８】１）コンピュータのランク

２）コンピュータのローカル負荷

３）負荷情報が伝送されるエッジまたはリンクを単位とした距離

４）子の情報を伝播する最後のノード（最後のノードフィールド）

負荷ベクトルの長さ、すなわち、ベクトルが含む項目の数はノードからノードに変化させることができる。

17

【0109】負荷情報は次のようにしてネットワーク上を配送される。

【0110】1．周期的に、各コンピュータはその負荷Lを抽出し、Lという新しい値を用いてその負荷ベクトルを分類し、負荷ベクトルをupdate_upメッセージに入れてその親に送信する。また、update_upメッセージはツリー保守のため動作可能であることを示すものとして用いられる。

【0111】2．ノードkからのupdate_upメッセージがノードmにより受信されると、

a．受信された負荷ベクトルの各項目の長さは1だけインクリメントされる。

b．受信するノードmから発信され、かつ、受信された負荷ベクトルの項目は全て削除される。

【0112】c．送信するノードkから発信された負荷ベクトルの項目は全て削除される。

d．ローカルベクトルおよび受信されたベクトルの両方に現れる項目（コンピュータ）の距離は比較される。

【0113】1．ローカルベクトル項目の距離が受信されたベクトル項目の距離より大きいか、あるいは等しい場合、ローカルベクトル項目が削除される。

【0114】2．受信されたベクトル項目の距離がローカルベクトル項目の距離より大きい場合、受信されたベクトル項目が削除される。

【0115】e．受信されたベクトルの項目の最後のノードフィールドは、送信するノードkになるようにセットされる。

【0116】f．受信されたベクトルの項目はローカル負荷ベクトルと併合される。

【0117】3．親はその新しい負荷ベクトルを、update_upメッセージが送信された子に、送信する。このメッセージはupdate_downメッセージであり、update_downメッセージは、その親が動作可能であることを示すものとしてツリー保守メッセージで再び用いられる。

【0118】4．update_downメッセージが到達すると、それを受信する子は受信された負荷ベクトルをステップ2で記載したのと同様の方法で処理する。

【0119】update_downメッセージが到達すると，メッセージを受信する子は受信された負荷ベクトルをステップ2と同様にして処理する。

【0120】update_downメッセージは、ツリー構造にリンクされている各ノードから、各ノードにより受信された情報を伝播する。よって、子はその親に＊

18

＊渡して情報を共用し、その親は情報を他の子に渡す。負荷情報をツリー内で上下に伝播することにより、別のサブツリーの標準より小さい負荷のノードに関する情報が任意のノードに到達することができる。負荷ベクトルのサイズが固定されているので、最低負荷状態のコンピュータの部分集合に関する情報を保持する。

【0121】負荷ベクトル内の負荷情報は、任意の特定のコンピュータの実際の負荷の近似のみである。というのは、その負荷情報が別のノードに到達するまで、負荷情報を発信するコンピュータの負荷は変化することができる。

【0122】「update間隔」は連続する負荷分散、すなわちupdate_upメッセージの時間間隔として規定されている。この時間間隔は負荷平衡の結果に対して大幅な影響を与える。値が非常に大きいため、他のノードの現在ノード負荷の情報が不正確になる。その情報が不正確なので、この情報に基づく移行判定結果は拒否が多くなる。値が非常に小さいので、すなわち、updateレートが大きいので、その方法のオーバヘッドが増加し、かつ、応答時間が低下する。そのupdateレート（1／update間隔）はローカルプロセスの発生レートに正比例し、かつ、このレート変化のように変化すべきである。さらに、考慮に入れる要素が他にもある。現在のノードのupdateレートは、その子からその親、あるいはその親からその子に対する情報伝播が減速されないことが確認されなければならない。現在ノードのレートが小さく、かつ子または親のレートが大きい場合に、情報伝播が減速されることになる。

【0123】そのため、中間の隣接ノードのupdateレートをまた考慮しなければならない。

【0124】ノードのeffective_rateはこれら2つの要素に基づきレート計算関数の結果として規定されている。本発明の本実施例では、次の関数が用いられる。まず、各ノードのeffectiveレートはローカルプロセスの誕生レートにセットされる。各ノードに対して、次の量が計算される。

【0125】

a）$rate_1 = local\_birth\_rate_p$ ローカル誕生レートpはスライディングタイムウィンドウに亘る平均として計算される。

【0126】b）pの全ての中間隣接ノードnに対して、

$$rate_2 = max\{effective\_rate_n\} / \alpha$$

である。αは次に示すコンピュータシミュレーション実験により決定される減衰係数である。

※【0127】pのeffective_rateは

$$effective\_rate_p = max\{rate_1, rate_2\}$$

である。

【0128】コンピュータシミュレーション実験では、1．4から2．0の範囲内のαの値は、負荷が均等に分散されたネットワークに対して良い結果が得られ、大き

い負荷の領域を有するネットワークに対して良い結果が得られる。

【０１２９】コンピュータが過負荷になると、余分の負荷を受け取ることができるコンピュータを示す第１項目に対するローカル負荷ベクトルを探索し、余分の負荷を受け取ることができるコンピュータに、１つ以上のタスクを直接転送する。負荷ベクトル内の目標コンピュータの負荷項目は更新され、負荷ベクトルは再分類される。目標コンピュータが何等かの理由で余分の負荷を受け取ることができない場合は、送信元が通知される。この場合、送信元は負荷ベクトルからその項目を削除し、次の項目を検索し、検索された項目を新しい目標として用いる。余分の負荷を受け取ることができるノードが１つもない場合は、移行は開始されない。

【０１３０】したがって、位置指定方策、すなわち、目標ノードを見付け出すことが、負荷ベクトルを分類するのに用いられる分類基準に依存する。負荷情報はホップしてツリー上を伝播されるので、ノードはさらに遠くなり、情報の精度が落ちる。そのため、この評価例では、負荷ベクトルは負荷および距離に基づき分類される。項目はそれらの負荷に応じてまず分類される。そして、等しい負荷を有する項目は距離の降順で分類される。

【０１３１】長い時間、軽負荷であるノードは、多くのノードの負荷ベクトル内の第１項目として現れる。このため、多くのノードはプロセスをそのノードに移行させて、そのノードを溢れさせる。このことがボトルネックとなる。この問題は２つのアプローチを用いて避けることができる。

【０１３２】等価ノードの集合を、負荷ベクトル内で、同一負荷および距離を有するノードとして規定することができる。例えば、負荷－１および距離２を有するノードは全て同一の等価集合内にある。この場合、分類基準により、同一等価集合内のノードは全てベクトルで群分けされることが確認される。負荷ベクトルが（ｕｐｄａｔｅ＿ｕｐまたはｕｐｄａｔｅ＿ｄｏｗｎメッセージにより）分散されるまで、ベクトル内の第１等価集合が置換される。このため、同一ノードが多くの負荷ベクトルに現れる確率が小さくなる。

【０１３３】さらに、まず、軽い負荷のノードがそれ自体の負荷ベクトルに最初に現れるといつも、カウンタが＊

＊項目に接続され、しかも、初期化され、常に零未満の軽負荷ノードの負荷に等しくされる。ベクトルが分散されるといつも、その項目がまだ第１である場合、カウンタがインクリメントされる。カウンタが正になる場合、その項目がそのベクトルから削除される。よって、軽負荷のノードを知るノードの数が制限され、ノード溢れの確率が小さくなる。

【０１３４】プロセス移行の開始は、ローカル負荷が検査される周波数に依存するとともに、ローカル負荷の値に依存する。ノード負荷ステータスを検査するか否かの判定は、選択された方策に依存する。可能な方策は数多くあり、次のようなものを含む。

【０１３５】１．周期的　　ローカル負荷はｕｐｄａｔｅ＿ｕｐメッセージを送信した後、試験される。すなわち、ローカル負荷抽出はｕｐｄａｔｅ＿ｕｐメッセージに同期される。

【０１３６】２．ドライブされる事象　　ローカル負荷はローカルプロセスが実行待ち行列に付加されるときはいつでも試験される。

【０１３７】コンピュータの過負荷をいつ宣言するかの判定、すなわち、過負荷マークをどこに置くかの判定は、その方法の実施に影響を与える。過負荷マークを余り低く置くと、移行が余りにも多くなり、応答時間が改善されない。過負荷マークを余り高く置くと、応答時間が低下する。過負荷マークの値は任意の特定のシステムに対して最適である必要があり、その値は、異なる負荷容量を有するネットワーク内のコンピュータに対するか、あるいは異なるジョブプロファイルに対して、異ならせることができる。最適な１つの値または複数の値を、所定の状態で、例えば、適正なコンピュータシミュレーション実験を利用するか、あるいは、試行錯誤のプロセスにより、確立することができる。

【０１３８】各ノードの構成ファイル内にランクを適正に割り当てて、ノードの制御されたクラスタ化を行うことができる。

【０１３９】例えば、ａからｋの１１個のコンピュータがある場合、次のように、４つのコンピュータのクラスタと７つのコンピュータのクラスタを形成する必要がある。

ａ　ｂ　ｃ　ｄ
第１クラスタ

ｅ　ｆ　ｇ　ｈ　ｉ　ｊ　ｋ
第２クラスタ

次のステップを実行してそれらのノードをランク付けする。

※【０１４０】１．各クラスタは別々にランク付けされる。例えば、

ｅ　ｆ　ｇ　ｈ　ｉ　ｊ　ｋ
５　６　０　１　２　３　４

をまず選択した場合、

ａ　ｂ　ｃ　ｄ
１　２　０　３

２．各クラスタを順番に取り上げ、前のノードの数だけ、各ノードのランクを増やす。例えば、第２クラスタ

ａ　ｂ　ｃ　ｄ
８　９　７　１０

ｅ　ｆ　ｇ　ｈ　ｉ　ｊ　ｋ
５　６　０　１　２　３　４

21

異なるランク付けファイルをコンピュータに割り当てることにより、クラスタが構成される。クラスタ内で最低ランクを有するコンピュータを除き、各コンピュータは、そのクラスタ内のノードのランクを含むが、そのクラスタ外のノードのランクを含まないランク付けリストを記憶する。その最低のランクを有するコンピュータは他のクラスタのノードを有するランク付けリストを記憶する。

【０１４１】そのため、クラスタのノードは他のノードを知らないので、相互接続されることになる。最低のランクを有するノードのみが別のクラスタのノードに接続されることになる。図１１は上記例のノードから形成されたサンプルツリーを示す。

【０１４２】ｃに対するこの最高数が３である場合、ａ，ｂ，およびｄの好ましい親リストはｃのみを含み、その結果、図１２に示すツリーが得られる。

【０１４３】ｃの好ましい親リスト内にｊを置くと、図１３に示すツリーが生成される。

【０１４４】本発明の実施例では、各ノードに対するランク付けファイルと好ましい親リストを賢明に選択することにより、クラスタ化が行われる。

【０１４５】以上、過負荷ノードによりタスクの転送が開始される送信元開始の負荷平衡方策を用い、本発明の実施例を説明したが、本発明は、受信先開始方策、すなわち、軽負荷のコンピュータが負荷の大きいコンピュータから自分自身にタスクの転送を開始する方策に、同様に適用可能である。

【０１４６】ネットワーク内のコンピュータを操作する方法と、改善された負荷平衡方策を用いてコンピュータのネットワークを操作する方法とを記載してきた。この改善された負荷平衡方策は拡張可能であり、フォールトトレラントであり、柔軟性があり、しかも、クラスタ化をサポートしている。よって、クラスタ化を、コンピュータの数が少ないネットワークからその数が非常に多いネットワークまでの利用に適合させている。

22

【０１４７】

【発明の効果】以上説明したように、本発明によれば、上記のように構成したので、より速く負荷を平衡させることができるという効果がある。

【図面の簡単な説明】

【図１】ツリー生成の要求および受信フェーズを示す流れ図である。

【図２】ツリー生成の要求および受信フェーズを示す流れ図である。

【図３】ツリー生成の要求および受信フェーズを示す流れ図である。

【図４】ツリー生成の要求および受信フェーズを示す流れ図である。

【図５】本発明に係るツリー生成の段階を示す図である。

【図６】本発明に係るツリー生成の段階を示す図である。

【図７】本発明に係るツリー生成の段階を示す図である。

【図８】本発明に係る実施例におけるツリー保守を説明するための説明図である。

【図９】本発明に係る実施例におけるツリー保守を説明するための説明図である。

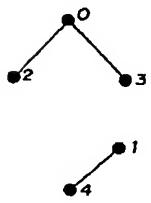
【図１０】本発明に係る実施例におけるツリー保守を説明するための説明図である。

【図１１】本発明に係る実施例におけるノードのクラスタ化処理を説明するための説明図である。

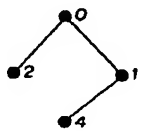
【図１２】本発明に係る実施例におけるノードのクラスタ化処理を説明するための説明図である。

【図１３】本発明に係る実施例におけるノードのクラスタ化処理を説明するための説明図である。

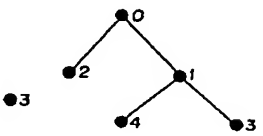
【図５】



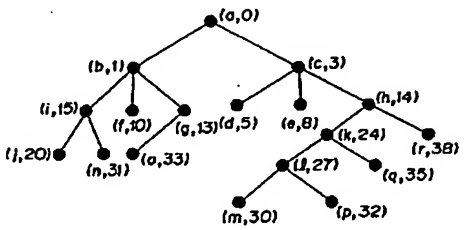
【図６】



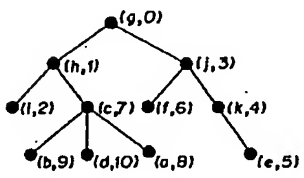
【図７】



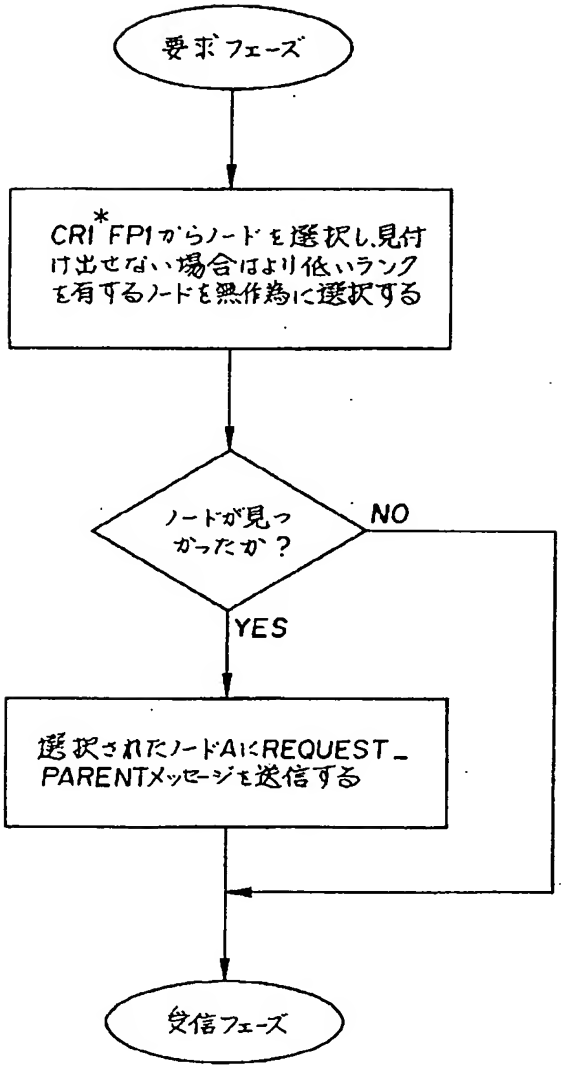
【図８】



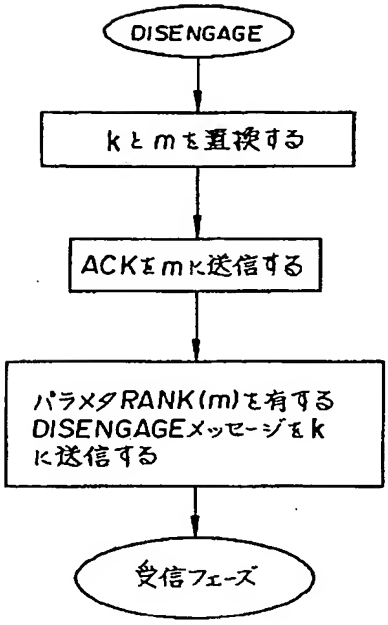
【図１２】



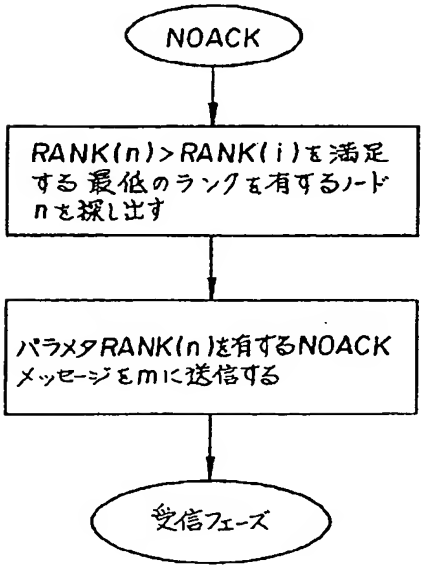
【図1】



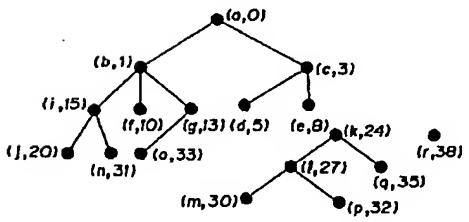
【図3】



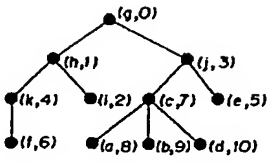
【図4】



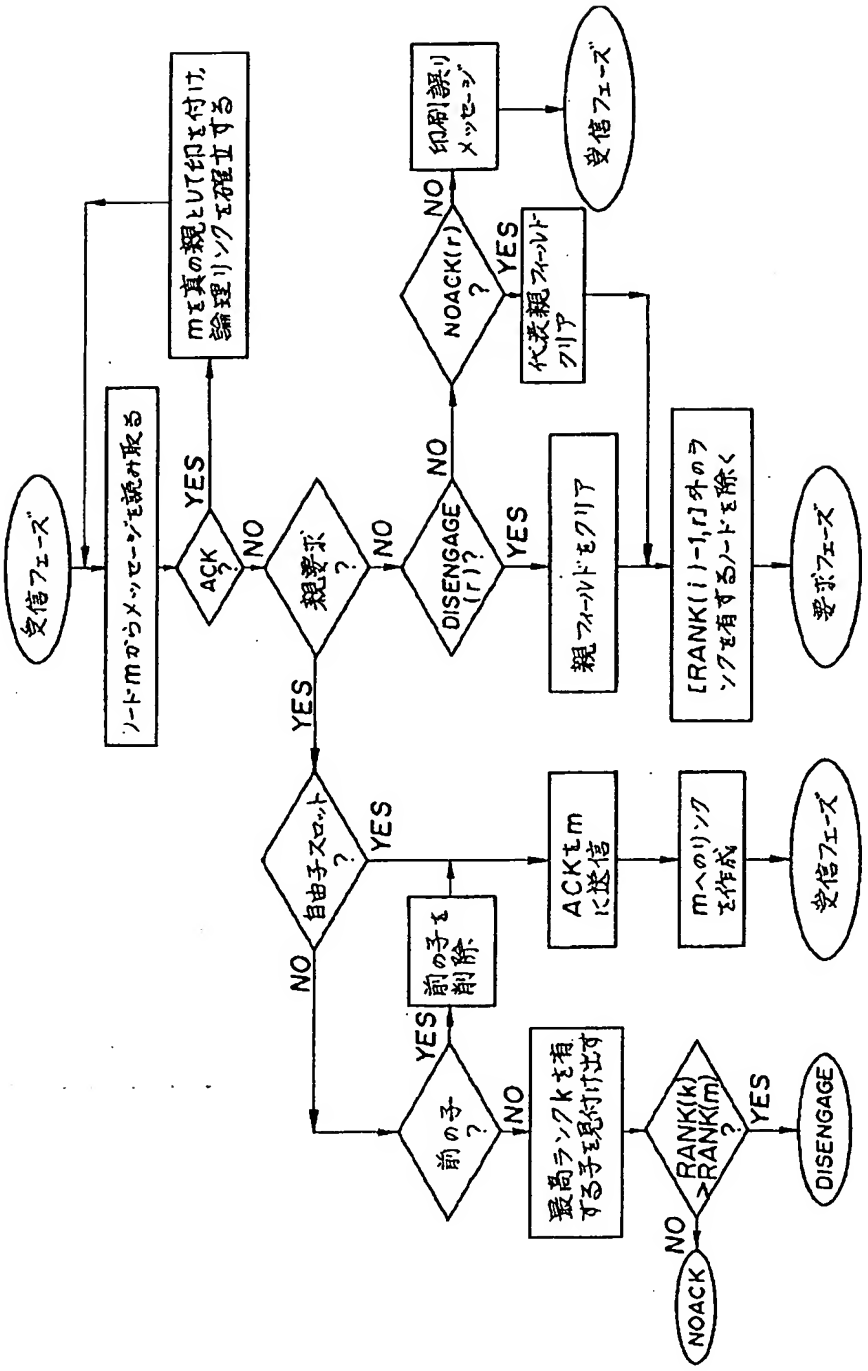
【図9】



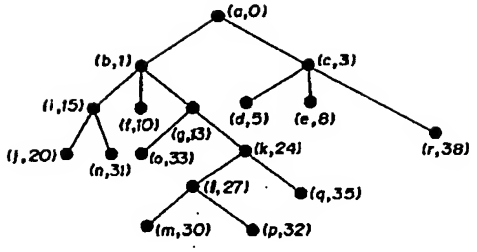
【図13】



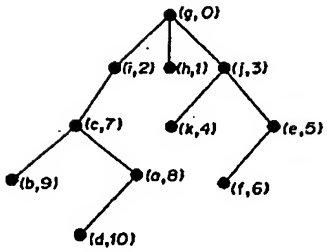
【図2】


受信フェーズ
ノードmからメッセージを読み取る
ACK?
YES
NO
mを真の親としてU印を付け、論理リンクを確立する
親要求?
NO
YES
DISENGAGE(r)?
NO
YES
NOACK(r)?
NO
YES
印刷誤りメッセージ
受信フェーズ
代表親フィールドクリア
親フィールドもクリア
[RANK(i)-1,r]外のランクを有するノードを除く
要求フェーズ
自由子スロット?
YES
NO
前の子を削除
ACKをmに送信
mへのリンクを作成
受信フェーズ
前の子?
YES
NO
最高ランクkを有する子を見付け出す
RANK(k)>RANK(m)?
YES
NO
DISENGAGE
NOACK

【図１０】



【図１１】



フロントページの続き

(72)発明者　モシェ　バッハ
イスラエル　ハイファ　トルンペルドール
ストリート　５エイ

(72)発明者　ヨセフ　モアティ
イスラエル　ハイファ　ハニタ　ストリート　68／56

(72)発明者　アブラハム・テバーマン
イスラエル　ハイファ　ハビバ　ライヒ
ストリート　46